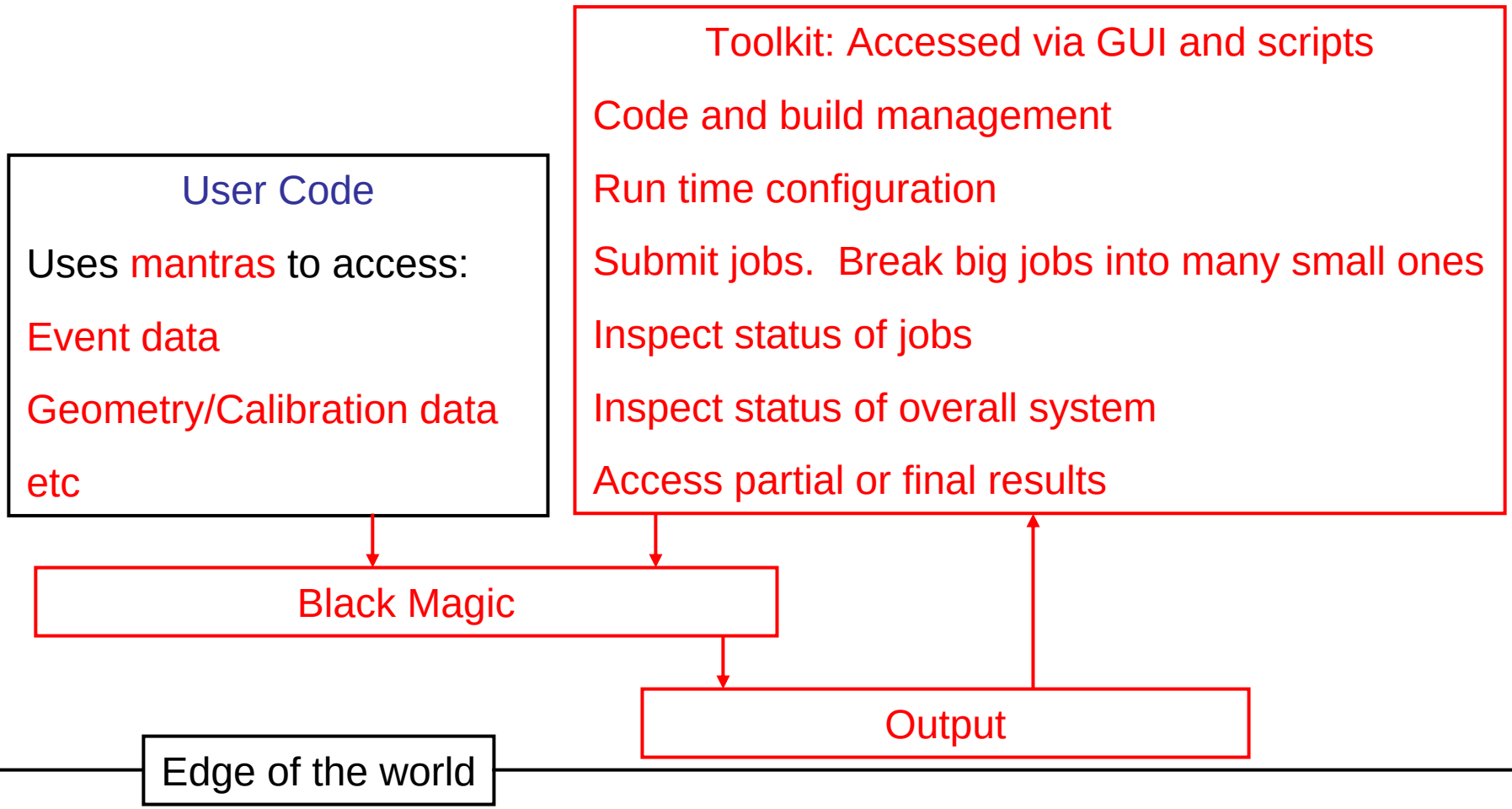
Offline Computing: Defining Some Components

Rob Kutschke, Fermilab

Offline Computing Computing Mini-Workshop

June 22, 2004

A Physicists' View



A Definition

- Calibration constants are almost never constant. Because of this, the term in standard usage is: conditions data.
- I will use both terms (trying to change all to use conditions data).

A Principle

- There needs to be a single authoritative source for every piece of information.
- There may need to be mirrors/replicas or caches to solve access issues.

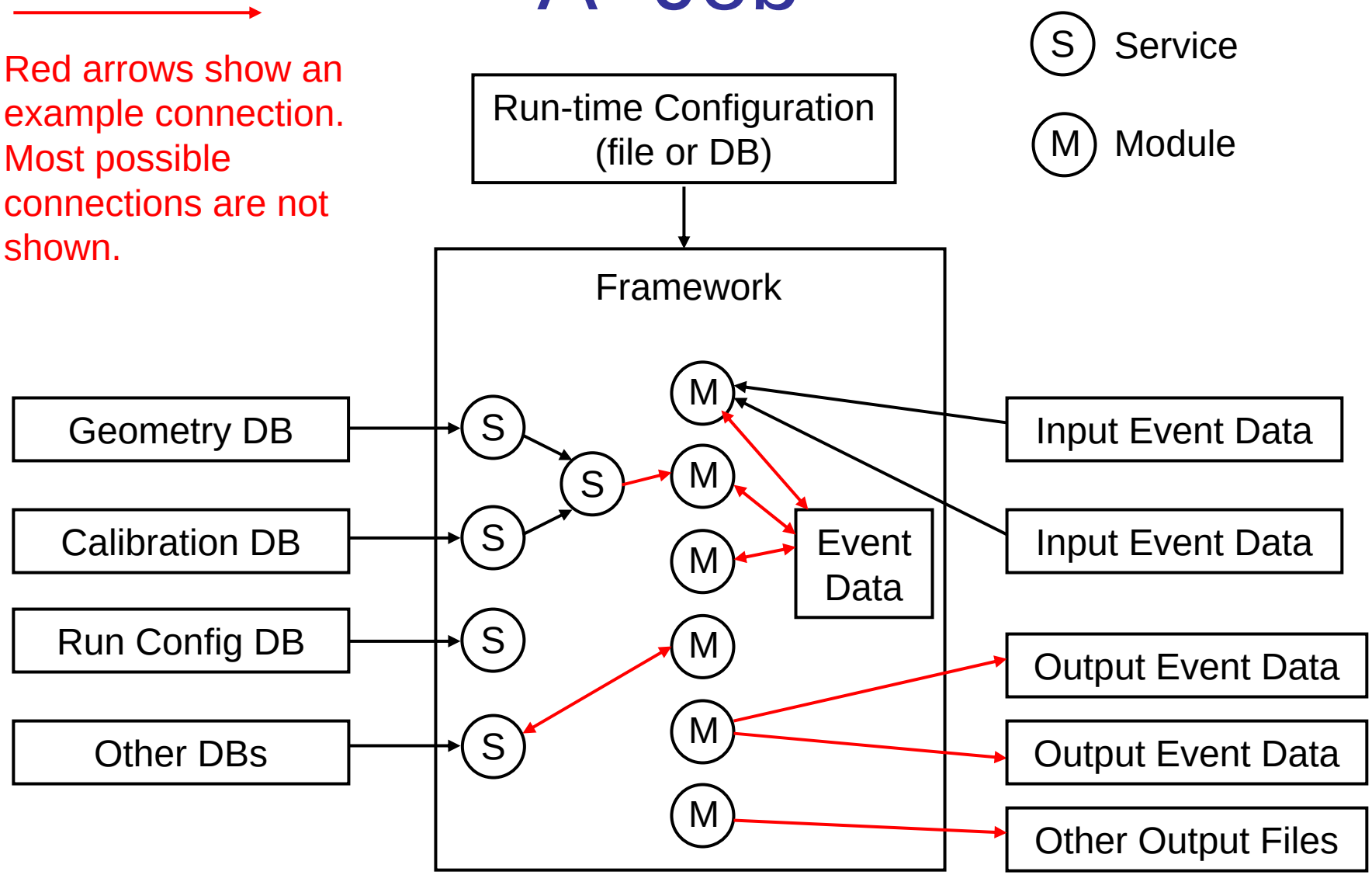
Define Infrastructure

- We need to build the infrastructure in which the physics software will live.
  - The Black Magic
  - The tools in the toolkit box.
  - The stuff that makes the mantras work.
- Some components are good candidates for Grid based solutions.
- I did not call this “framework”.
  - Reserve this for something else..

A Requirement Unique to BTeV

- Infrastructure must be small and fast enough to be used in L2/L3 trigger yet powerful enough for general use.
  - Physics code must not need to know about its execution environment.
- Part of the answer:
  - Zero overhead principle.
    - No overheads for resources you do not use.
    - Sidebar: Need to be able to ask for a group of resources in a single request.

A “Job”


Red arrows show an example connection. Most possible connections are not shown.
S Service
M Module
Run-time Configuration (file or DB)
Framework
Geometry DB
Calibration DB
Run Config DB
Other DBs
S
S
S
S
S
M
M
M
M
M
M
Event Data
Input Event Data
Input Event Data
Output Event Data
Output Event Data
Other Output Files

Comments on Previous Slide

- A job:
  - Is the primitive unit which is managed by the infrastructure.
  - Can also be run standalone on a laptop if all resources are locally available.
- Physics code lives within the modules.
  - Small pieces may be inside the services?
- Goal: computing professionals will write the non-physics code.

Comments …

- A single job might, for example,
  - Process data:
    - Raw data … DST and all in between.
  - MC event generation and detector simulation.
  - Inspect/modify at calibration information.
  - May do several of the above.
- Some tasks will require coordinated, multiple runs of one or more such jobs.

Services

- The interface to give modules access to
  - Geometry, calibration, error loggers, profiler, debugging tools …
- Services may have some awareness of their execution environment but physics modules must not (IO modules may be aware).
- Services and modules must be plug replaceable.
- The module/service model allows a fast start and a flexible, robust evolution path.
  - Initially services may be very simple things and later they may become fully Grid aware.
  - Specialized/fast versions of some for use in L2/L3 trigger.

Comments on the “Job” Slide

- Modules may talk to services.
- Modules may communicate with each other only via the event or, indirectly, by signaling the framework.
- Run-time configuration starts only the services and modules that are needed.
- Modules can signal the framework that something interesting has happened.

Framework

- After the initialization phase, it is just a fast, lightweight dispatcher.
  - For each event, call the “process event” method of requested modules.
  - For each new run, call the “end run” method and then the “new run” method of requested modules.
  - And so on for “end of job” and for many other possible things that can happen.
  - For each thing, only call the relevant modules.

Framework (II)

- All jobs, MC gen, MC sim, reco, skim, analysis, should have a uniform interface for:
  - Run time configuration.
  - How to read/write event data.
  - How to access geometry and conditions data.
  - Error loggers, random numbers,
  - Methods to insert user code.

What Do We Give Up?

- Scheduling and resource allocation are layered:
  - Above job level, done by the infrastructure.
  - Within a job, is done by within the framework.
  - No chance to cross-optimize.
    - My guess is that this is not important.

Granularity

- One key is getting the granularity correct.
  - How are events “chunked”.
  - How are calibration data “chunked”.
  - Scheduling and resource allocation.
  - Provenance: There are many levels:
    - Data catalog / File / Collection of Tracks / Track
- Choosing extremes of granularity is very likely to be wrong.

Message Logger

- Uniform API for informational messages, warnings, errors, severe errors.
  - Distinct from, but related to, exceptions and “aborting” events.
- Statistics generated at end of run, end of job or on request.
- Possible to route different severities differently, ie highest priority might be routed real time to RTES.

Random Numbers

- Need to maintain sequences of random numbers across many jobs. Even across GRID jobs.
- Independent streams of random numbers within a single job.
  - Keep all else the same but try different model of, for example, hit generation.
- Want an integrated solution.

Geometry and Alignment

- Three distinct ideas:
  - Nominal geometry.
  - Time dependent alignment information.
    - This is a subset of the conditions data.
  - The fully calibrated geometry.
    - Most users will want only this and not the other two.
- I think that these end up as three distinct pieces of code. May have very different technologies behind them.

When Do I Read Geometry?

- Constructor of my class?
- At start of job?
- At an arbitrary time?
- At beginning of a run?
  - I think we should choose this.
  - We will want to do luminosity weighted averages over many runs.
- Let's do this from the start.

When Do I Read Geometry (II)

- I would prefer that the framework call me when the geometry changes.
- Don’t want to figure out on my own that it is time to update the geometry.
- Should geometry system return pointers smart enough to know that their target has disappeared?

What is a Run?

- A period of constant detector and Tevatron configuration.
- An artifact created for bookkeeping convenience.
  - Constants may only change on run boundaries.
- An index into the geometry/conditions databases.
- The least of:
  - A period of clock time.
  - A number of events.
  - A file size.
  - Other?

Is a Run too Big?

- Some calibration constants change much more frequently than others. Is it OK to have a run change every 10 minutes?
  - ECal healing?
  - Average luminosity?
  - Beam position?
- Convenient to define a “Run segment”.
  - Just a bookkeeping convenience.
  - Do we want this?

What a Run is Not

- I think it is a bad idea to encode other information in the run number.
  - Run 2 is MC for B0 -> pi pi
  - Run 3 is MC for B0 -> J/Psi K0s.
  - We should have other methods to store this.
- I would like the run to be strictly an index into the conditions data.

Geometry Wish List

- Should be able to hold several geometries at once.
  - Simulation/Reco to study alignment.
    - Simulate with one alignment
    - Reconstruct with another
    - Derive correction.
  - Pre-loading of next calibration during L2/3.
    - To smooth the disruptions at calibration transitions and reduce the needed pipeline depths.

Run Time Configuration

- Things we want to be persistent.
  - Which detector components are turned on, what cut or quality levels are set.
  - Want to be able to say “generate the standard MC but turn off the ECAL and muons”.
- Things we do not need to be persistent.
  - Debug levels, enable/disable diagnostic outputs, input file.

Hit Formats

- Packed raw data.
- Unpacked:
  - Electronic address + raw ADC.
  - Logical address + raw ADC.
  - Logical address + semi-calibrated ADC.
- Clusters of any of the above.
- Cluster associated with a track with final corrections applied (angle of incidence).
- Can we develop a language to keep these ideas distinct?

Hit Formats (II)

- Would like MC hits to be usable as are data hits, where appropriate.
- Also want MC hits to have (optional) MC truth info.
  - Will be discussed more a little later.
- Please include a plan for MC truth when you make the hits. Critical of debugging pattern recognition and fitting algorithms.

Events Model Etc

- Items below are distinct but related.
- Event model.
  - Describes an event in memory.
- Persistence model(s).
  - Describes how events are stored/read to/from disk.
- Data Catalog.
  - Describes where to find files.

Event Model

- Event: a header plus containers of objects.
- Each container identified by a provenance.
  - Is this the right granularity for provenance?
- Adding a container to the event needs to be fast and safe.
  - Use appropriate safe pointers.
- Once a container is added to the event, it becomes read only.
  - Good for audit trail.
  - Need a mechanism to add info later.

Event Wish List

- If I modify, for example, the track header, I don’t want to recompile all code which references an event but which does not use tracks.
- If I want to add a new type of thing to the event, I don’t want to have to recompile all code which references an event.

Linking Between Containers

- Examples:
  - Tracks to showers
  - Tracks to vertices
  - Hits to tracks
  - MC hit to MC truth
  - Reco track to MC truth
  - Cluster to hits
- Would like an integrated solution for all.

Linking Between Containers (II)

- Example track shower matching:
- Event contains a container of tracks and a container of showers.
- Match code must not modify either tracks or showers.
- Instead match code adds a new container of “track-shower match” objects.
- Easily allows multiple algorithms to be tested. Each algorithm adds its own container.

Linking Between Containers (III)

- I know of three solutions:
  - POOL, a CERN product in development.
  - D0's solution, in use.
  - CLEO's solution, in use.

Linking Between Containers (IV)

- What to do about the following?
- I have a long list of tracks.
- I find a set of vertices.
- I want to keep all tracks in secondary vertices or not in a vertex and discard the other tracks.
- How do my vertex to track links survive this process?

DAQ in This Event Model

- Here is one scenario.
- On each L2/3 node a separate task maintains an input event pool.
- The IO module can get a pointer to the raw data from this task.
- The object added to the event can be an appropriate safe pointer to the raw data.
- The destructor of this object can talk to the first task and tell it to free the memory.

IO Subsystem

- Want multiple input and multiple output streams.
  - Input:
    - MC Signal file.
    - Charm background file.
    - Other background file.
  - Output
    - Multiple skims/splits in one job.
    - Different content on each skim.

Databases

- We will use databases to hold lots of information.
- Will need a lot of work on things such as mirroring/replicas, caching to ensure prompt access by many jobs, especially in the trigger.
- Production databases probably do not need to be tightly coupled to the offline code. We just need snapshots.

Unit Safety

- I know of two mechanisms.
  - G4
  - SIUnits
- Should we adopt one throughout?
- Use G4 within G4 and SIUnits elsewhere?
- Forget it all?

Other Thoughts

- Are there good ideas which are too hard to teach or too hard to convince our colleagues to adopt?
  - Thread safe code?
  - Exception safe code?
- Do we want sub-event parallelism?
- Where is boundary between OO and whatever people want?

Boiler Plate

- Do not optimize for rare use cases at the expense of common use cases.
  - A use case should say how often it occurs.
- Be wary of external products for which we do not have access to the sources.
  - Can be trapped by incompatible versions:
    - DOE security mandate: version n+1 of product A.
    - But product B does yet have a version which is compatible with product A, version n+1.
    - Data taking stops.

Boiler Plate …

- Overly ambitious, poorly done tools can be much worse than nothing at all.
  - Be realistic about what we can do with the available resources.
  - Re-use solutions developed elsewhere.
- The evolution of a traditional HEP farm to a full blown data grid is a natural path with many stable islands along the way.
  - Island hopping sounds like a good plan.

Final Thoughts

- Our job is to get the physics done.
- We need to be Grid aware and ready to exploit mature Grid tools.

Other Things

- Constrained fitting package, vertex and mass fits.
- B Field service.
  - How to integrate G4, and all reco parts?
- Combiner engine.

Working Material

End Users' View in More Detail